STAP
Smart Technologies Academic Press

Journal of Cyber Security and Risk Auditing

https://www.jcsra.thestap.com/

A Hyperledger Fabric Blockchain Framework for Secure Authentication and Flow Verification in Hierarchical SDN

Stefina Macwan[1], Sailesh Suryanarayan Iyer[2*], Rami Shihab[3], Amir Alqatish[4*]

[1] *Ph.D Scholar, Rai School of Engineering, Rai University, Ahmedabad, India*
[2] *Research Supervisor, Rai School of Engineering, Rai University, Ahmedabad, India*
[2]*Professor and Principal, Narnarayan Shastri Institute of Technology-Institute of Forensic Sciences and Cyber Security, Jetalpur, Ahmedabad, India*
[3] *Vice-Presidency for Postgraduate Studies and Scientific Research, King Faisal University, 31982, Al-Ahsa, Saudi Arabia*
[4] *Deanship of Development and Quality Assurance, King Faisal University, 31982, Al-Ahsa, Saudi Arabia*

ARTICLE INFO

Article History

Received: 27-01-2026

Revised: 25-04-2026

Accepted: 15-05-2026
Published: 19-06-2026

Vol.2026, No.2

DOI:

Corresponding author. Email:
drsaileshiyer@gmail.com and aalqatish@kfu.edu.sa

Orcid:

https://orcid.org/0000-0002-0619-9829

ABSTRACT

Software Defined Networking (SDN) has transformed network management by separating the control plane from the data plane. This enables logically centralized control and network-wide policy enforcement. Centralization is especially important in hierarchical SDN architectures used in large-scale, geo-distributed data centers. Our work addresses two major threats in hierarchical SDN: Cluster Membership Exploits (CME), which allow unauthorized controllers to join the controller cluster, and malicious flow manipulation, in which compromised controllers inject unauthorized flow entries. These attacks can threaten the cluster's integrity, enable unauthorized system access, and disrupt tenant segregation. To tackle these problems, we have developed a security architecture that uses Hyperledger Fabric and hierarchical OpenDaylight controllers. This architecture leverages Hyperledger Fabric's endorsement policies, immutable ledger, and permissioned transaction processing to enforce controller membership validation and flow integrity. All flow installations require approval from both the root controller and the corresponding regional controller. With the help of decentralized consensus mechanisms and immutability of blockchain, our solution offers secure authentication and authorization of all connections within the cluster, as well as pre-installation verification and tamper-evident logging of flow-related control-plane operations.

Keywords: Software Defined Networks, Blockchain Technology, Cluster Membership Exploit, Data Centers.

How to cite the article

1. Introduction

Software-Defined Networking is changing how large data centers are configured and maintained in the established networks. These networks are simplified when the control plane, which makes decisions, is separate from the data plane, which transfers data to hardware like switches or routers. This separation improves policy enforcement and service orchestration while giving the controller a global view of the network.

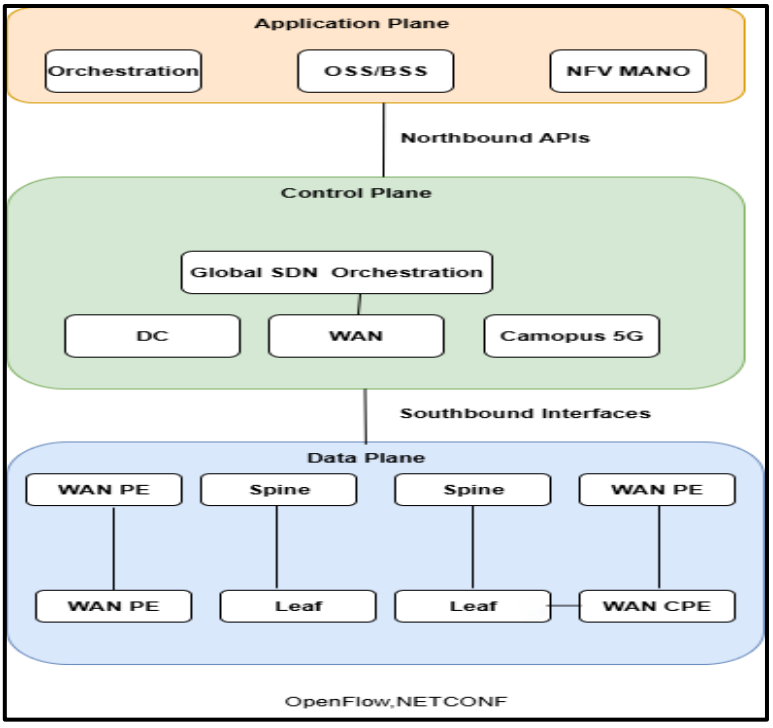


Figure 1. Software-Defined Network Architecture

With the rapid growth of cloud services and geographically distributed enterprises, Software-Defined Networking (SDN) as demonstrated in Figure 1, is adopting a tiered method of implementation, where regional controllers in such hierarchical networks managing specific regions or data centers. The root controller makes sure that every policy is never compromised. Hierarchical controller architectures are commonly used in large-scale SDN deployments to manage geographically distributed infrastructures. It makes security difficult, though, because every regional controller can be attacked, and the question of the safety of the network arises.

In Figure 2, the application layer has orchestrators and automation tools, the control layer has hierarchical controllers (root and regional), and the data layer has switches that handle tenant traffic. The figure also shows two important ways to attack hierarchical SDN: (1) Cluster Membership Exploits (CME), where a malicious controller enters the cluster; such attacks enable rogue or unauthorized controllers to infiltrate a network cluster by circumventing security measures and threatening the integrity of the entire network. For example, it was discovered that the Authentication, Authorization, and Accounting (AAA) component of OpenDaylight up to version 0.19.3 had a flaw that enabled a rogue controller to pretend to be an offline peer without having access to the complete cluster configuration information.

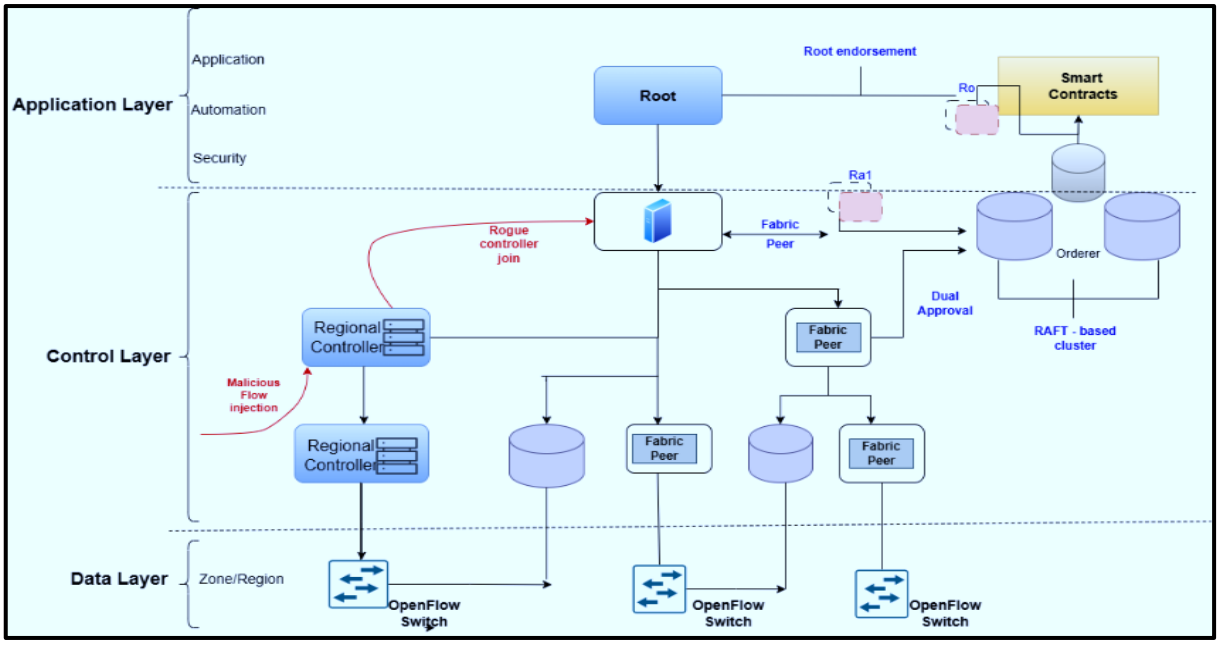


Figure 2. SDN layered architecture in a large-scale data center

Once within the cluster, these rogue controllers can alter network operations, interfere with services, and steal important data without permission, all of which can have detrimental effects on operations and finances. The Cybersecurity and Infrastructure Security Agency (CISA) has identified this vulnerability as a CVSS base score of 9.1 (critical) to highlight the severity of the issue in network systems that are based on controller-mediated security controls, and (2) Flow Table Manipulation where a compromised controller introduces unauthorized flows. By performing these attacks, attackers can route traffic by altering flow entries in SDN switches. These alterations can cause information leakage, unavailability of services, and decreased trust in the management system of the network. These attacks have already exposed SDN infrastructures to danger in the real world. Researchers demonstrated that a faulty OpenFlow [12] controller can use CME to enter into a cluster and modify the settings of switches. The BlackNurse DDoS [33] attacks had utilized improperly configured SDN flow tables on ISP backbones, resulting in service failure across a large area. Similarly, the IoT deployments based on SDN have been targeted by variant versions of the Mirai malware [34], where controllers are controlled to redirect or drop traffic at scale. These incidents demonstrate that hierarchical SDN [24] designs can be highly useful to the operations, as well as become highly attractive to attackers.

To perform controller-to-controller authentication, the standard SDN controllers, including OpenDaylight [26], rely on static X.509 certificates. Installing these certifications is a requirement, yet it does not give you any dynamism or fine-tuning control of who should be included or excluded in the cluster. In case of certificate theft or misuse, rogue controllers may enter the cluster without any notice, and this may jeopardize the integrity of the control plane. In addition, the majority of the existing SDN systems do not offer tamper-resistant logs of control events. In case some controller is compromised, logs may be deleted or modified, and there is no way to know what happened and do a forensic analysis.

To mitigate these vulnerabilities, we consider the integration of hierarchical SDN control planes with Hyperledger Fabric, a permissioned blockchain platform that provides decentralized consensus, immutable ledgers, and programmable chaincode for the enforcement of fine-grained policies. Each SDN controller in our architecture has a Fabric peer. Smart contracts (also called chaincode) guarantee that only members who have been approved by the root can join the cluster. This means that any new regional controller needs to obtain permission from the root before joining. In the same way, flow installation requests must have approval from both the regional controller who proposed the flow and the root controller prior to it can be carried out. All membership and flow transactions are inscribed in Fabric's ledger in an immutable manner. Consequently, to ensure robust security against CME and adverse flow manipulation, our method utilizes Fabric's decentralized trust and endorsement mechanisms.

Besides preventing unauthorized controllers from accessing the system, it hinders compromised regionals from independently installing flow and produces records of control operations that can be cryptographically verified. This is essential to preserve tenant segregation, ensure operational safety, and uphold a high security standard for hierarchical SDN control planes employed in big multi-tenant data centers.

2. Related works

Studies represented in Table 1 on Blockchain-SDN integration have sought more to decentralize the control plane, enhance controller-to-controller trustworthiness, and guarantee the imparting of mission-critical security events. Initial SDN security testing [1,5], the traditional SDN implementations are identified to have structural weaknesses, such as the use of a logically centralized controller, poor controller authentication, and the lack of verifiable mechanisms to check the integrity of deployed flow rules. These papers indicate that there is a requirement of ongoing, cryptographically implemented membership validation and strong mechanisms to overcome malicious or unqualified malicious flow insertion vulnerability that cannot be overcome by standard SDN architecture.

Single-Controller Vulnerabilities and Flow-Rule Vulnerabilities: Single-controller SDN designs leave the network to disastrous compromise in case the controller is compromised in some manner. In earlier literature, the security of the malicious controller or the infected northbound application has been shown to inject arbitrary entries of flow to allow the redirection of traffic, blackholing, or hijacking of the session without any built-in protection. Despite the fact that multi-controller SDN frameworks reduce the risks of single-point failures, they also present novel trust and consistency threats. Experiments compiling distributed and locality-aware SDN control models [3, 2] show that it is not enough to distribute the control logic, unless membership authentication and inter-controller trust anchoring is performed with stringency, making the attack surface large.

IoT-Based blockchain-SDN Mechanisms: Another similar line of research, is to use blockchain on IoT-centric SDN environments. Researchers used MaxSMT-based [4] validation to ensure SDN-IoT interactions, which improves the correctness of decisions but is applied to limited IoT ecosystems and not high-throughput data-center fabrics. In other literature, lightweight blockchain-based trust-score systems are presented to authenticate IoT devices [8].

Access-Control Security and Northbound Interface: Decentralized authorization frameworks have been used to provide security to the SDN Northbound Interface (NBI). B-DAC [7] system uses Hyperledger Fabric to impose fine-grained, policy-based access control of NBI applications in the form of immutable records of application actions. Although B-DAC heavily intensifies the security of the applications, it does not tackle hierarchical SDN, or topologies and flow-level dual endorsement.

Flow Validation at the Network Edge: Blockchain-assisted validation of flow rules at the Mobile Edge Computing layer is introduced by the Edge BC-SDN [35] architecture. Although it assures that only legal flow requests are granted at the edge, its blockchain agents (BCAs) work only on switch-level validation and reward-based procedures. The characteristics required to completely secure large-scale SDN deployments controller membership, hierarchical coordination, and dual-level flow permission between parent and child controllers are not covered.

Blockchain-based IDS and Immutable Log Management: New frameworks such as SmartSecChain-SDN [36] combine multi- model machine-learning IDS with Hyperledger Fabric in order to get tamperproof logging, flow profiling, and application-aware QoS classification. Although such systems possess a high accuracy of runtime detection and high auditability, they lack verifiable controller identity, hierarchical trust enforcement, or multi-party flow-endorsement.

Table 1. Comparison of Key Related Works on SDN Security and Blockchain Integration

Paper & Year	SDN Architecture	Blockchain Use	Membership Enforcement	Flow Verification	Application Focus
Nisar et al., 2020 [1]	General SDN	None	Identifies need	Identifies need	Survey

Lin et al., 2018 [2]	Single-controller	None	No	Highlights flow attacks	Attack modeling
Hossein et al., 2019 [3]	Multi-controller	None	No	No	Architecture overview
Bringhenti et al., 2022 [4]	SDN–IoT	MaxSMT	Partial	Policy-based	IoT networks
Alqobaty & Ahmed, 2024 [6]	Data center SDN	Blockchain for scalability	Partial	No	Controller scaling
Duy et al., 2022 (B-DAC) [7]	General SDN	Hyperledger Fabric (NBI security)	API-level	No	Northbound access control
Deng et al., 2024 [8]	IoT SDN	Lightweight blockchain	Trust score-based	No	Resource-constrained IoT
Krishnamohan, 2020 [15]	SDN	Blockchain replication	No	No	Controller decentralization
Abou El Houda et al., 2019 [10]	SDN	Blockchain-assisted DDoS mitigation	No	No	DDoS defense
Xie et al., 2022 [19]	SDN	None	No	No	Control-plane attack taxonomy
LiBSCOM-AS (Garg et al., 2025) [37]	Multi-AS SDN	Blockchain for collaborative DDoS mitigation	Partial (AS-level)	No	Inter-domain security
Hu et al., 2021 (Edge BC-SDN) [35]	SDN–IoT with MEC	Smart-contract-based flow validation	No	Yes (edge chaincode)	IoT/edge flow integrity
SmartSecChain-SDN, 2025 [36]	SDN (Ryu/ODL)	Hyperledger Fabric	No	No	ML-based IDS + QoS

Proposed Work	Hierarchical ODL SDN	Hyperledger Fabric (endorsement + chaincode)	Root-endorsed on-chain registration	Dual verification (root + regional)	Multi-tenant data centres

All these studies investigate controller security, access control, scalability, and blockchain integration, but none of them examine hierarchical SDN designs, enforce root-endorsed membership, and require dual endorsements for flow approvals at the same time, as our research does. Using Hyperledger Fabric, our solution uniquely integrates these parts to directly fill in the gaps found in surveys like those by [1, 5,17].

3. System design

The primary objective of the proposed system is to ensure the security of hierarchical SDN control planes that are deployed across large-scale, multi-site data center networks by enforcing cryptographically verifiable controller membership and flow integrity. The architecture is designed to address two primary threats: the introduction of unauthorized controllers into the cluster (Cluster Membership Exploits, CME) and the injection of unauthorized flow entries by malicious or compromised controllers. To accomplish these goals, Hyperledger Fabric [32] utilizes its decentralized trust, unalterable ledger, and programmable chain code [31] endorsement policies. These offer evident security guarantees that aren't offered by standard SDN implementations.

Overall Architecture: The proposed system comprises three tightly integrated layers: the ordering layer, the control layer, and the data plane layer. A high-level view of the design is illustrated in Figure 3.

Ordering Layer: Ordering layer is executed as a Hyperledger Fabric ordering service based on RAFT. This layer keeps the non-mutable ledger which logs the events of membership of the controllers and validated flow transactions. Orderer cluster involves several nodes that make up the orderer cluster and carry out the following functions:

- Transaction ordering
- Block creation
- Peer-to-peer ledger synchronization

All authenticated control-plane operations in this layer are written as blockchain transactions, which make them tamper-resistant to audit and consistent in all global states.

Control Layer: Provides hierarchical SDN control plane of one root controller and a number of regional controllers. The root controller happens to be the major policy authority. It accepts memberships and grants flows. The local switches are controlled by regional controllers and assist in flow entries. Both controllers are linked to a Fabric peer through an instance of OpenDaylight (ODL) through the ctrl-mgmt channel. This allows them to read the shared ledger and to implement chaincode logic.

Data Plane Layer: This layer consists of switches utilizing OpenFlow or NETCONF, distributed across data center locations in several regions. Every switch is operated by its respective regional controller. Regional controllers utilize protocols such as BGP-LS or RESTCONF to establish a connection with the root controller. This enables the root controller to maintain a coherent, overarching perspective of the network topology.

Membership Approval Workflow: The architecture shown in Figure 4 ensures that any controller joining the SDN cluster has root-endorsed membership to facilitate cluster formation. The workflow proceeds as follows:

a) A regional controller sends a join request by executing the ***Join()*** function in the membership chaincode (membership_cc) and supplying its unique ID and public key.
b) The Fabric endorsement policy stipulates that the root controller's Fabric peer must endorse the transaction. Requests lacking this endorsement are automatically rejected.

c) Upon confirmation, the new controller's membership is permanently registered in the Fabric ledger. This ensures that only approved controllers are permitted to join the cluster. This process formalizes membership acceptance as define in Eq. (1):

$$\textbf{Accept}(M_r)= \begin{cases} 1, \text{ if } E_{root}(M_r) \text{ is valid,} \\ 0, \text{ otherwise,} \end{cases} \quad (1)$$

Where M_r is the membership request and $E_{root}(M_r)$ the root's endorsement.

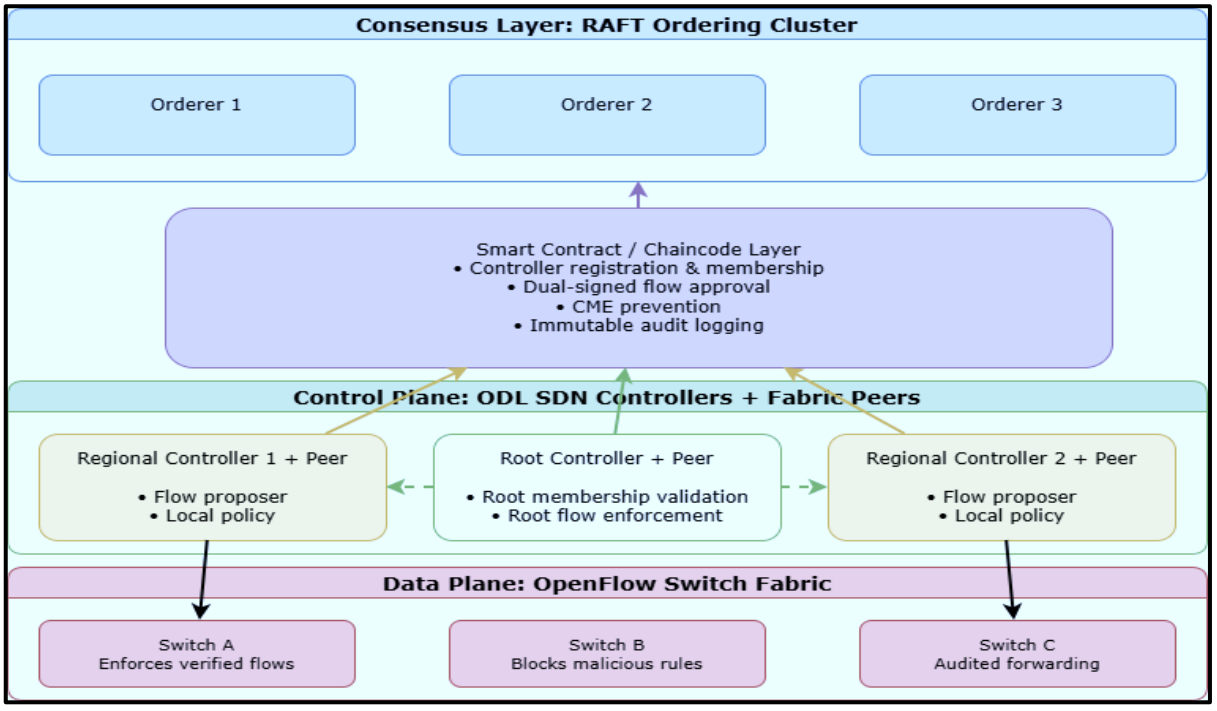


Figure 3. Proposed Architecture

Flow Verification and Installation: To prevent unauthorized or policy-inconsistent flows, every proposed flow by a regional controller must undergo two evaluations:

a) The regional controller utilizes the flow's match-action fields to generate a cryptographic hash H (Fr) and subsequently executes the ***Push()*** function within the flow chaincode (flow_cc).
b) The endorsement policy requires that the transaction receive approval from both the regional controller's peer and the root controller's peer. If either endorsement is absent or fraudulent, the flow proposal is rejected.

The ODL-Fabric integration only starts OpenDaylight's MD-SAL Flow Service to program the flow on specified switches when both dual endorsement and ledger commit are successful, which is shown in Eq. (2).

$$\textbf{Install}(F_r)= \begin{cases} 1, \text{ if } E_{root}(H(F_r)) \text{ and } E_r(H(F_r)) \text{ are valid,} \\ 0, \text{ otherwise,} \end{cases} \quad (2)$$

Where $E_{root}(H(F_r))$ and $E_r(H(F_r))$ are the root and regional endorsements of the hashed flow $H(F_r)$

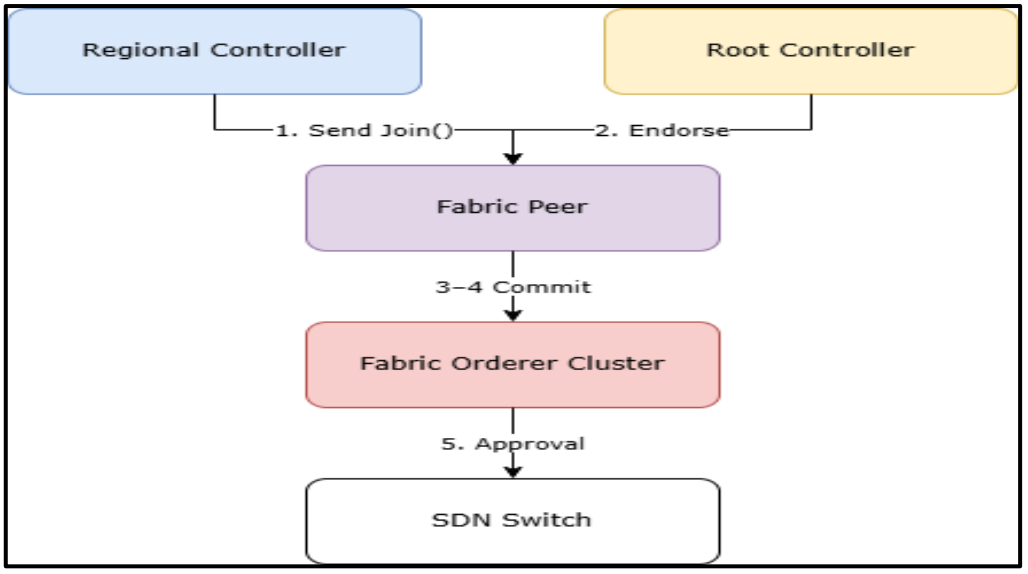


Figure 4. Membership Approval Workflow

Chaincode Components: There are two Hyperledger Fabric chaincodes that set the rules for security**:**

a) Membership Chaincode (***membership_cc)*** handles ***Join()*** requests, makes sure that only root-approved members can join, and keeps track of authorized controllers.

b) Flow Chaincode ***(flow_cc)*** controls ***Push()*** requests and needs two approvals before validated flows may be stored in the ledger.

c) The Fabric peer network makes sure that both chaincodes' endorsement policies are followed on the ctrl-mgmt channel.

Algorithm 1 Secure Membership + Dual-Signed Flow Verification

//Phase 1: Controller registration (join): A controller submits a registration request which is initially marked as **PENDING** until validated by the root controller.

Procedure RegisterController(controllerID, role)

if controllerID ∈ ControllerTable then
reject request

ControllerTable[controllerID] ← {role, PENDING}

record registration event

//Phase 2: Root-approved membership (CME prevention): Only the **root controller** can activate or revoke controller membership.

Procedure ApproveController(rootID, targetControllerID, decision)

require ControllerTable[rootID].role = ROOT
require ControllerTable[rootID].status = ACTIVE

if targetControllerID ∉ ControllerTable then
reject request

```
    if decision = APPROVE then
       ControllerTable[targetControllerID].status ← ACTIVE
    else
       ControllerTable[targetControllerID].status ← REVOKED

    record membership update
```

//Phase 3: Flow proposal by regional controller: Regional controllers propose flows but **cannot install them directly**.

```
Procedure ProposeFlow(controllerID, flowID, switchID, match, actions)

    require ControllerTable[controllerID].role = REGIONAL
    require ControllerTable[controllerID].status = ACTIVE

    if flowID ∈ FlowTable then
       reject request

    FlowTable[flowID] ← {controllerID, switchID, match, actions, PENDING}

    record flow proposal
```

// Phase 4*: *Root-Endorsed Flow Validation: This step enforces **dual authorization** between regional and root controllers.

```
Procedure ApproveFlow(rootID, flowID, decision)

    require ControllerTable[rootID].role = ROOT
    require ControllerTable[rootID].status = ACTIVE

    if flowID ∉ FlowTable then
       reject request

    if decision = APPROVE then
       FlowTable[flowID].status ← APPROVED
    else
       FlowTable[flowID].status ← REJECTED

    record flow decision
```

// Phase 5: Flow Installation: Only flows marked **APPROVED** are installed in the network.

```
Upon FlowStatusChanged(flowID)

    if FlowTable[flowID].status = APPROVED then
       install flow on target switch
    else
       discard flow request
```

Threat Model: Ensuring the security and integrity of the hierarchical SDN control plane is critical, particularly multi-tenant deployments where distributed controllers manage traffic flows across geographically separated switches. In such environments, the compromise of a single controller can potentially impact the entire network infrastructure. The proposed ODL–Fabric framework mitigates these threats through blockchain-based controller membership validation and flow integrity enforcement. This section illustrates the threat model of the proposed architecture that has attacker capabilities, trust assumptions and its mitigation mechanisms.

Attacker Capabilities: Under hierarchical SDN with Hyperledger Fabric, we have an adversary that we believe is capable and can attack the control plane by tapping into the weaknesses of controller membership, flow programming, and inter-controller communications. Particularly, the opponent is supposed to have the following capabilities:

Unauthorized Controller Join (Cluster Membership Exploit, CME): An attacker may attempt to register a malicious controller instance within the SDN control plane without proper authorization. If successful, the attacker could manipulate network policies, inject malicious flow rules, or gain persistent control over switch operations. A controller membership request C_i is considered malicious if the root endorsement is invalid or if the controller does not possess a valid Fabric-issued certificate showed in Eq. (3).

$$E_{root}(C_i) = 0 \vee Cert(C_i) \notin L_{valid} \quad (3)$$

Where:

- $E_{root}(C_i)$ denotes the root controller endorsement of controller C_i
- $Cert(C_i)$ represents the certificate associated with controller C_i
- L_{valid} is the list of valid Fabric-issued certificates

Malicious Flow Injection: The attacker can also seek to add flow entries to the data plane that are unauthorized or otherwise inconsistent and bypass control-plane verification. Such attacks may redirect traffic, drop packets (blackholing), or bypass security policies, which is mention in the Eq. (4).

$$Install(Fr) = \begin{cases} 1, \text{ if } E_{root}(H(F_r)) \wedge E_r(H(F_r)) = 1, \\ 0, \text{ otherwise,} \end{cases} \quad (4)$$

Unauthorized CLI Invocation: In permissioned blockchain systems such as Hyperledger Fabric, network participants may interact with the blockchain using Fabric CLI tools. An attacker may attempt to exploit this interface by invoking transactions using forged identities or invalid MSP credentials.

A transaction request T is rejected if the submitted identity cannot be verified by the Membership Service Provider (MSP), defined in Eq. (5):

$$Verify_{MSP}(T_{identity})=0 \quad (5)$$

Where *VerifyMSP* represents the cryptographic identity validation performed by Fabric peers.

This mechanism prevents attackers from executing unauthorized membership updates or flow transactions through CLI-based access.

Man-in-the-Middle (MITM) Attacks: The adversary may intercept, eavesdrop on, or modify control-plane communications between controllers and Fabric peers over RESTCONF or gRPC channels. Such attacks can compromise transaction authenticity, lead to flow tampering, or allow the spoofing of controller endorsements. Mitigation relies on cryptographic guarantees of mutual TLS to ensure confidentiality and integrity, which is mention in the Eq. (6):

$$Integrity(M) = Confidentiality(M) = 1 \; \forall \; M \in TLS \quad (6)$$

For all valid TLS messages M.

3.1 Trust Assumptions

1. **Root Controller (C_{root})**: Fully trusted; responsible for approving all membership and finalizing flow endorsements.

2. **Regional Controllers (C_{reg})**: Trusted for local operations but potentially compromised. Dual endorsement ensures flows cannot be applied without root validation.
3. **Fabric Orderers and Peers**: Follow RAFT consensus. Majority assumed honest; at least 2-of-3 orders must agree on block commits.
4. **Network Links:** RESTCONF and gRPC channels secured with mutual TLS. Attackers cannot forge or modify messages without valid certificates.
5. **Ledger Integrity:** Once entities are accepted into the Fabric ledger, they cannot be changed and any effort to overwrite or modify the past records is blocked by the blockchain protocol.

Table 2. describes the threats can be exploited at each layer of SDN and its mitigation mechanism.

Table 2. Threat Mitigation Summary

Threat	Layer	Mitigation Mechanism
CME	Control	Membership chaincode, Fabric Certificate, Root endorsement
Malicious Flow	Control/Data Plane	Flow chaincode, dual endorsement, hash verification
Unauthorized CLI Invocation	Blockchain Layer	MSP identity verification and endorsement policy enforcement
MITM	Control/Peer Comm	Mutual TLS authentication

4. Implementation

Experimental Testbed: The proposed architecture was set up on a private cloud with six virtual machines (VMs). Each virtual machine operated on Ubuntu 24.04 LTS and was equipped with 4 virtual CPUs and 8 GB of RAM. The testbed was configured as follows:

a) VM-1 (**Root Controller):** Hosted an OpenDaylight (ODL) instance connected to a Hyperledger Fabric peer. This Fabric peer was responsible for authorizing new members and facilitating flow transactions.
b) VM-2 (**Regional Controller - Region A):** Established an ODL instance and a Fabric peer, both within the same blockchain network.
c) VM-3 **(Regional Controller - Region B):** Configured identically to VM-2, incorporating an ODL instance and a Fabric peer inside the same region.
d) VM- 4, VM-5, VM-6 **(Fabric Orderer Cluster):** Establish an alternative Hyperledger Fabric orderer nodes that safeguards the ledger and manages membership and transaction flow.

Each controller VM (VM-1 to VM-3) possesses a Fabric peer and the control-plane SDN software (ODL). This meant that they could directly take part in the processes of validating on-chain membership and endorsing dual flows. To safeguard control and fabric traffic from unauthorized network access, all virtual machines (VMs) communicate via an isolated management.

Configuration of the OpenDaylight Controller: Each ODL instance was configured with the following:

a) Southbound Plugins: NETCONF [28] and OpenFlow [27] for communication with programmable switches.
b) Northbound Interfaces: MD-SAL and RESTCONF, which facilitate integration with external applications and enable programmability.
c) Hierarchical Mounting: The establishment of a hierarchical control topology as illustrated in Figure 2 is achieved by BGP-LS mount-points between regional and root controllers.
d) Akka-based clustering was implemented by regional controllers to enhance intra-regional resilience. However, inter-regional control was exclusively reliant on hierarchical relationships that were enforced through BGP-LS.

Configuration of Hyperledger Fabric: Hyperledger Fabric v2.5.13 was employed to construct the blockchain network.

a) Fabric Orderers (RAFT-based,) that guarantees consensus and immutability.
b) Three fabric peers (root and regional) are taking part in the ctrl-mgmt channel.
c) Fabric's cryptogen facilitates the secure distribution of cryptographic identities and certificates.

d) The confidentiality and integrity of membership and flow transactions were preserved by all Fabric nodes, which communicated over gRPC secured by TLS.

ODL-Fabric Bridge: A custom OSGi bundle (odl-fabric-bridge) was developed in Java to integrate OpenDaylight with Hyperledger Fabric. It provided:

a) Listening capability for ODL's MD-SAL Flow Added events.
b) gRPC-based Fabric Gateway connection to submit membership and flow transactions.
c) RESTCONF endpoints to automate membership registration during controller startup.
d) Blocking mechanisms to prevent flow installation until Fabric confirms transaction commits.

Automated and validated deployment: For repeatability and repeatable experimental deployments, all system components were automated with Ansible playbooks. Installation of software dependencies, configuration of hierarchical SDN controllers, deployment of Hyperledger Fabric components, and blockchain network chaincode initialization were part of the automation pipeline. These validation tests show that the proposed architecture successfully implements security requirements, preventing Cluster Membership Exploits and illegal flow installations n hierarchical SDN systems. Automated workflow included these steps:

a) Software Environment Setup: Automated installation of OpenJDK 21, Maven, Docker Compose, and all system dependencies on all controller VMs.
b) Set up OpenDaylight instances on Root and Regional controllers and configure BGP-LS mount-points to establish hierarchical links.
c) Initializing the multi-node RAFT-based orderers after generating and distributing cryptographic materials (certificates, keys) for all peers and the Fabric orderers using Fabric's cryptogen tool.
d) Automatic initialization of Fabric peer containers on each controller VM and deployment of membership (membership_cc) and flow (flow_cc) chaincodes. The playbooks included Fabric CLI scripts to install, approve, and commit both chaincodes on the ctrl-mgmt channel.
e) Endorsement Policy Configuration: Chaincode endorsement rules require root approval for membership transactions and dual endorsements (root and regional) for flow transactions, per Section 3's security design.
f) The integrated system was validated using Mininet-emulated OpenFlow switches on each regional controller.
g) Experiments included: Membership requests from untrusted regional controllers without Fabric certificates or root endorsements are unauthorized. The membership chaincode consistently refused these requests, validating the root-approved cluster membership policy.
h) Malicious Flow Injections: Programming regional controller flows without dual endorsements. The flow chaincode blocked unendorsed transactions from reaching the data plane.
i) Ledger Auditability: The Fabric ledger constantly and immutably recorded all permitted membership and flow transactions, allowing post-event verification and forensic auditing.

4.1 Experimental Metrics and Statistical Validation

To make our evaluation of the proposed blockchain-assisted SDN framework effective, rigorous, and reproducible, we have chosen carefully performance metrics and workloads during experiments to reflect the security effectiveness and performance of the SDN control plane. All the experiments were performed on a dedicated virtualized infrastructure, which involves several virtual machines with Ubuntu, onto which OpenDaylight controllers and Hyperledger Fabric components were deployed. In contrast to all simulation-based assessments, this design enables the behavior of the system to be monitored in a realistic setting, such as Fabric peers, RAFT ordering services, and real controller-peer interaction.

4.2 Selection of Performance Metrics

The set of evaluation metrics was selected to gauge the effect of blockchain verification integration to the SDN control plane without degrading the operational efficiency. The metrics that were taken into account were the following:

a) Blockchain transaction latency, which measures the time required for a flow verification request submitted by the OpenDaylight controller to be endorsed, ordered, and committed to the Hyperledger Fabric ledger. This metric directly captures the processing overhead introduced by blockchain-based validation.
b) CPU utilization, which quantifies the computational overhead introduced by Fabric peer operations such as chaincode execution, endorsement validation, and RAFT consensus coordination.
c) Memory utilization, which measures the additional memory consumption introduced by Fabric components, including peer containers, endorsement caches, and the CouchDB state database.

d) Security enforcement effectiveness measured with a set of controlled attack experiments such as Cluster Membership Exploit (CME), malicious flow injection, unauthorized CLI invocation and attacks by the Man-in-the-Middle. These experiments show that the framework is able to discourage compromise of the control-plane.

Collectively, these metrics present a holistic perspective of the performance of systems and security assurances in the proposed architecture.

4.3 Selection of Experimental Workloads

Experimental workloads were simulated to be representative of actual SDN control-plane conditions of multi-tenant data center settings. Flow-injection rates were adjusted between 100 and 1000 events per second (eps) to approximate the varying controller workloads in the conditions of moderate to high network traffic. These are values that are usually used in the SDN performance analyzes and they enable checking of how the proposed framework can be scaled when the transaction demand is high. The architecture tested security experiments to monitor the consistency of the defense mechanisms applied to security through numerous attempts of attacks. Specifically:

- 50 attack attempts were performed for Cluster Membership Exploit and malicious flow injection scenarios
- 20 attempts were performed for unauthorized CLI invocation attacks
- 25 attempts were performed for Man-in-the-Middle attack simulations

These repeated experiments will give enough observations that will assess the soundness of the security enforcement mechanisms.

4.4 Statistical Reliability

In order to enhance statistical reliability, individual experiments of performance were replicated several times within the same system conditions in the virtual machine environment deployed. The data on blockchain transaction latency, CPU use, and memory use are reported as the means of repeated measurements made during these runs. The evaluation results are a dependable measure of the proposed architecture in terms of security effectiveness, scalability, and operational overhead by integrating experimental repetitions that are controlled and in a realistic multi-node, virtualized deployment.

5. Results and Analysis

A set of specific experiments was carried out to evaluate the security and effectiveness of the proposed hierarchical SDN framework that is incorporated with Hyperledger Fabric. These tests modelled realistic attack scenarios and fault conditions of the system in order to assess the effectiveness of the system in protecting the integrity of flow and implementing membership controls.

5.1 Security Analysis

Enforcement of Cluster Membership: We simulated Cluster Membership Exploit (CME) attacks by sending membership join requests from controllers that didn't have valid Fabric-issued certificates or root endorsements. The membership chaincode turned down all 50 of the unlawful join requests. This demonstrated that only controllers with Fabric-signed certificates that the root peer had approved were allowed to join the SDN cluster. In order to simulate, utilizing one of the following attack techniques, unauthorized controllers tried to register themselves, which is highlighted in Table 3:

Table 3. Cluster Membership Exploit (CME) attacks

Attack Scenario	Description
Invalid Certificate	Controller submits a join request without a valid Fabric-issued certificate
Missing Root Endorsement	Controller submits a request without root peer endorsement
Forged Identity	Controller attempts to reuse a valid identity with modified certificate fields

Each attack scenario was sent a join request through the Fabric Gateway API, and the join attempt was emulated. The results confirm that Cluster Membership Exploits are effectively avoided by applying Hyperledger Fabric to the SDN control plane. The malicious controllers cannot avoid verification of their membership, as identities of the controllers are cryptographically bound to the Fabric certificates and authenticated by the chaincode execution endorsed by the root. Unlike traditional SDN architectures, where membership to controllers is default or unsecured-api, the proposed architecture has immutable, and auditable membership records, which are enforced on the blockchain ledger. Every event that is joined by controllers is thus permanently logged hence post event auditing and forensics can happen.

Flow Integrity Verification: We tried making RESTCONF PUT requests directly to the southbound OpenFlow plugin of ODL but not receiving the appropriate blockchain endorsements to ascertain whether malicious flow injection would be achieved. The flow chaincode rejected all proposals at the flow without any endorsee in 50 tests; thus, all unverified flows could not be placed on the data plane. Unlike in the traditional SDN architecture where membership of controllers is enabled by default or unsecured API membership, the proposed architecture has immutable and auditable membership records, which is imposed on the blockchain ledger. All join events with controllers are thus logged and post event auditing and forensics can be done. Every malicious request tried to install entries in the flows that contained invalid match-action rules like:

- Redirecting traffic to a malicious host
- Dropping legitimate traffic
- Modifying forwarding paths

Defense against Unauthorized CLI Invocation: In order to test the resilience of the proposed architecture to identity spoofing attacks even more, we performed experiments against the Hyperledger Fabric command-line interface (CLI). Transactions are also invoked by invoking Fabric peer CLI in permissioned blockchain systems like Hyperledger Fabric. When an attacker tries to compromise a valid controller through fabricated cryptographic identities or false MSP credentials, they can also seek to make unauthorized transactions to the network. The suggested framework can address such attacks by enforcing Membership Service Provider (MSP) identity checks, cryptographic signature checks, and endorsement policies that are implemented by Fabric peers. This experiment determines whether a hostile party is able to circumvent these measures and make a successful attempt to make unauthorized membership or flow transactions. In order to emulate identity spoofing attacks, people tried to invoke the Fabric peer CLI with invalid or fake MSP directories. The 20 unauthorized CLI transactions were rejected at the endorsement stage because the signatures could not be verified.

Resilience of Man-in-the-Middle: To test how the proposed ODL+Fabric architecture would resist against Man-in-the-Middle (MITM)-attacks, we have performed experiments whereby an adversary was attempting to intercept and alter traffic between the OpenDaylight controllers and the Hyperledger Fabric peers. Control-plane communication integrity is also of great importance in SDN deployments since controllers use sensitive data, including membership transactions, flow verification requests, and ledger queries. When an attacker manages to intercept or modify these messages, he might end up injecting malicious transactions or changing flow policies or disrupt controller coordination. The given architecture will help to prevent this threat by applying mutual Transport Layer Security (mTLS) to all the communications between the controllers, Fabric peers, and orderers. This mechanism ensures that communication endpoints verify each other in terms of cryptographic certificates and then connect. In order to replicate the MITM attacks, a proxy server was mounted between the OpenDaylight controller and the Fabric peer. The proxy tried to interfere with and alter gRPC traffic carried across port 7051, the default port of communication of Fabric peers.
The attack attempts included the following scenarios, which is described in Table 4:

Table 4. MITM Attacks

Attack Scenario	Description
Message Interception	Proxy captures gRPC packets between controller and peer
Message Modification	Proxy attempts to alter transaction payloads
Certificate Spoofing	Proxy attempts to impersonate a Fabric peer using fake TLS certificates

All 25 MITM attempts were unsuccessful, as the TLS handshake failed whenever the proxy attempted to impersonate a valid peer or modify encrypted messages.

5.2 Ablation Study

In order to measure the role played by each element of the proposed framework, we performed an ablation study whereby a single security mechanism was disabled. This enables us to examine the contribution made by each of the modules towards the total security assurances of the system.

Four configurations were evaluated:

1. **Baseline SDN** – hierarchical SDN without blockchain validation.
2. **Membership Verification Only** – blockchain verifies controller membership but does not validate flows.
3. **Flow Verification Only** – blockchain validates flows but does not enforce controller membership.
4. **Full Framework** – both membership verification and flow validation mechanisms enabled.

Each configuration was tested against the attack scenarios defined in the threat model.

Table 5. Average Blockchain Transaction Latency

Configuration	CME Prevention	Flow Injection Prevention	MITM Prevention	Avg Blockchain Transaction Latency (ms)
Baseline SDN	✗	✗	✗	N/A
Membership Verification Only	✓	✗	✓ (TLS)	128.6
Flow Validation Only	✗	✓	✓ (TLS)	138.9
Full Framework	✓	✓	✓	146.9

To evaluate the contribution of individual security mechanisms, we conducted an ablation study by selectively disabling controller membership verification and blockchain-based flow validation. Table 5 summarizes both the security coverage and the average blockchain transaction latency observed across the tested flow-injection rates. The default SDN setup does not use blockchain transactions and thus there is no associated blockchain commit latency. The full structure has the longest average latency of all the blockchain-enabled structures since it combines membership verification and flow verification. Nevertheless, the mean latency can be effectively controlled within the range of operation of control planes.

a. Blockchain Transaction Latency Evaluation

To measure the performance overhead due to the inclusion of blockchain validation into the SDN control plane, we measured the transaction commit latency at various injection rates of flows. The time interval between the placement of a flow verification transaction by the OpenDaylight controller and its successful commitment to the Hyperledger Fabric ledger after peer endorsement, RAFT-based ordering and ledger persistence is referred to as transaction latency.

For comparison, we also consider the baseline ODL-only configuration, which installs flow rules directly on switches without invoking blockchain validation. Moreover, we also make reference to the SmartSecChain-SDN framework, the values of the transaction latency are taken out of the original research.

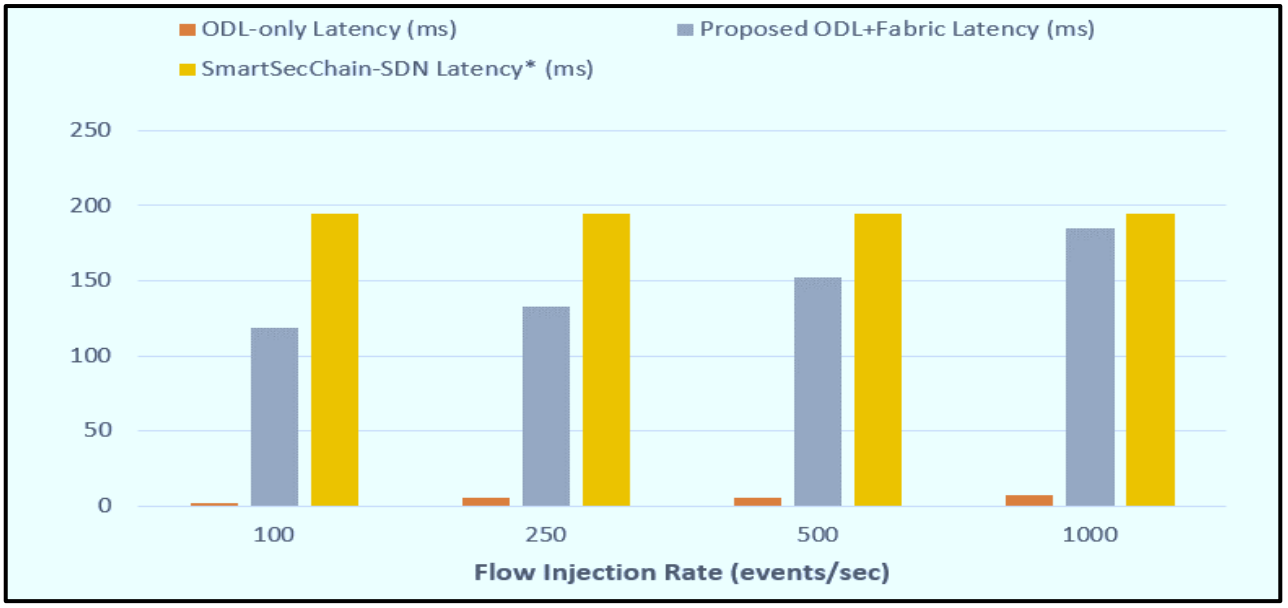


Figure 5. Blockchain Transaction Latency

The ODL-only configuration has the shortest latency, as in this case, flow rules are also installed directly on the controller, without blockchain verification or consensus work. Nevertheless, this setup offers quick installation of rules, but lacks any protection against malicious or unauthorized rule injection since it does not offer any way of verifying the authenticity of the controller or integrity of flow programming requests. Conversely, the overhead associated with ODL+Fabric architecture is more because it has a blockchain validation pipeline. Every flow verification request is entered into the Fabric network as a transaction, which is executed by a chaincode, verified by peer endorsements, ordered by a RAFT protocol, and stored in ledgers, before the controller continues with enforcing rules. These added validation checks add additional latency to transactions over and above that of the base setup. Notwithstanding this overhead, the performance indicates that the suggested architecture has steady and foreseeable behavioral patterns in relation to latency at varying injection rates of flows. The transaction latency increases in a consistent range with the increase in the control-plane workload as a result of an increased demand on the ordering service and the endorsement pipeline, which is highlighted in Figure 5. This conduct signifies that the blockchain implementation does not cause much instability or exponential growth in latency with augmented control-plane traffic. As a reference, the SmartSecChain-SDN architecture documents blockchain transaction times of between 134.2 ms and 228.4 ms based on the size of the block and block-batching approach applied in Hyperledger Fabric. In concurrent submission conditions of parallel transactions, the research [36] records a mean transaction process time of about 194.5 ms of a transaction.

The higher transaction latency in SmartSecChain-SDN is largely attributed to its architecture, which integrates machine-learning-based intrusion detection models and extensive security-event logging into the blockchain layer. Each detected event triggers additional processing steps, including ML inference, alert generation, and detailed audit logging, which increases the overall blockchain transaction overhead. In contrast, the proposed ODL+Fabric framework focuses specifically on controller membership validation and flow integrity verification, avoiding the additional computational overhead associated with ML-based intrusion detection pipelines. As a result, the architecture achieves lower and more predictable blockchain transaction latency while still ensuring strong security guarantees for the SDN control plane.

Overall, these results indicate that integrating Hyperledger Fabric into the SDN control plane introduces manageable latency overhead while significantly improving control-plane trust, flow integrity verification, and tamper-evident auditability.

b. Comparison of CPU Utilization

Figure 6 indicates the CPU usage at varying flow-injection rates. ODL-only configuration has an average of CPU occupancy, increasing by 15.8% at 100 eps to 33.1% at 1000 eps, which is due to the RESTCONF message handling and OpenFlow

southbound operations. The proposed ODL+Fabric configuration shows only a modest CPU overhead. This slight increase results from chaincode execution and the computational overhead required for RAFT block replication.

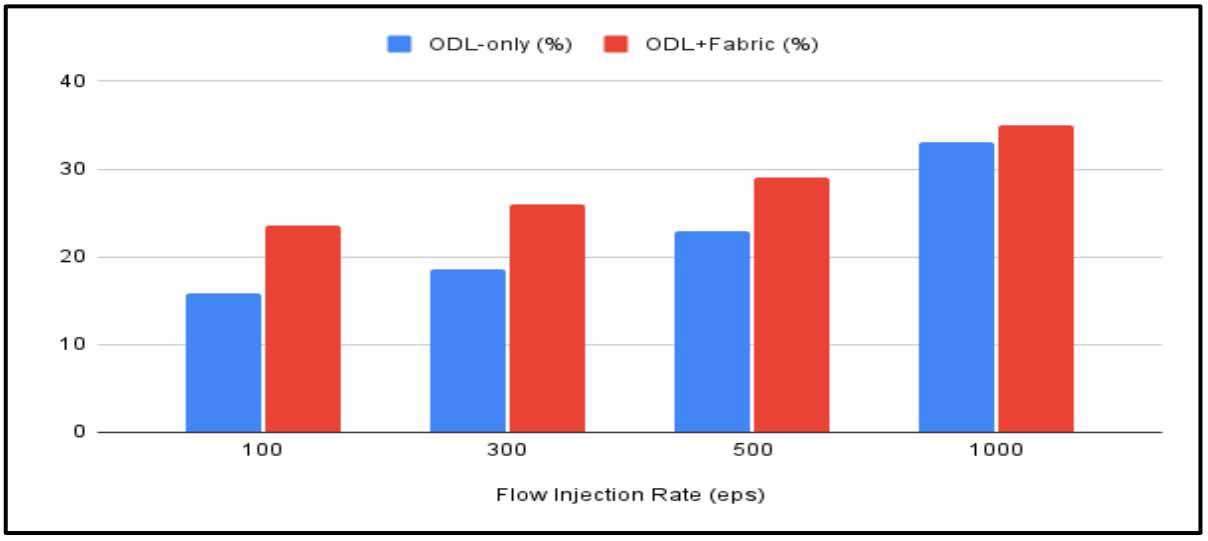


Figure 6. CPU Usage

SmartSecChain-SDN does not provide numerically comparable CPU measurements because its overhead is dominated by executing multiple ML models. Literature represents significantly higher CPU usage in ML-driven SDN systems, especially during real-time classification and blockchain logging.

c. Memory Utilization Analysis

To compare the system memory use for ODL-only, ODL+Fabric, and SmartSecChain-SDN, Figure 7 is given. The ODL-only controller uses between 620MB and 760MB of RAM, depending on the flow rate, which gives an efficient memory footprint. The proposed ODL+Fabric architecture and architecture requires a bit more memory in between 720MB and 880MB because of additional parts: CouchDB state database, peer containers and endorsement caches. However, memory consumption is below 1 GB at all load levels which is extremely resource efficient for a blockchain-backed SDN system.

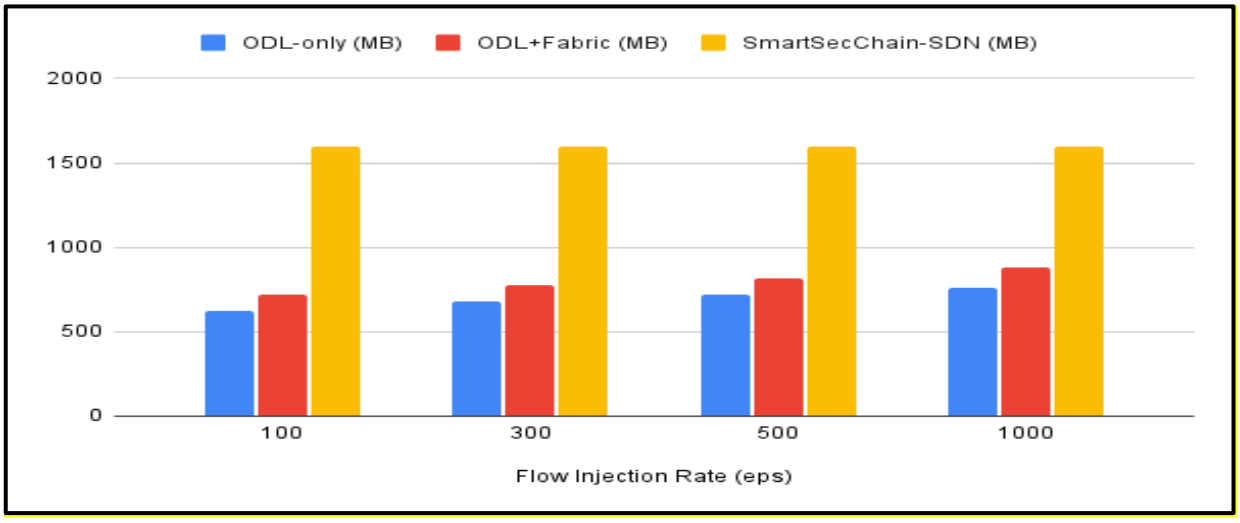


Figure 7. Memory Utilization

SmartSecChain-SDN on the other hand demands much larger memory budgets. ML inference engines in combination with Fabric logging infrastructure usually consume more than 1.6 GB in moderate load scenarios and higher volumes are reported in IDS-heavy scenarios.

d. Computational Overhead and Time Complexity Analysis

In addition to empirical measurements of latency, CPU utilization, and memory consumption, it is important to analyze the computational complexity and processing overhead introduced by the proposed blockchain-assisted SDN control-plane verification framework. This analysis is particularly relevant for multi-tenant data center environments, where multiple tenants simultaneously generate flow programming requests through their respective controllers.

The proposed architecture performs two main verification operations: controller membership validation and blockchain-based flow verification. The computational complexity of these operations depends on the number of tenants, controllers, and Fabric network components involved in transaction validation.

Let

T = number of tenants in the data center
C_t = number of controllers associated with tenant t
N_p = number of endorsing Fabric peers
N_o = number of ordering service nodes
F_t = number of flow requests generated by tenant t

Controller Membership Verification: Each tenant controller must first register with the control plane through the membership chaincode, which verifies the controller identity using Fabric-issued certificates and validates the authorization record stored in the blockchain ledger.

The cryptographic certificate verification requires a constant-time operation as follows Eq. (7):

$$T_{cert} = O(1) \tag{7}$$

The membership lookup in the blockchain state database depends on the number of registered controllers and can be modeled as Eq. (8):

$$T_{lookup} = O(\log(\sum_{t=1}^{T} C_t)) \tag{8}$$

Thus, the computational complexity for controller membership validation as follows in Eq. (9):

$$T_{membership} = O(\log(\sum_{t=1}^{T} C_t)) \tag{9}$$

Since the number of controllers per tenant remains relatively small in hierarchical SDN deployments, the overhead of this operation remains minimal.

Flow Verification Complexity: In a multi-tenant SDN data center, each tenant may generate multiple flow programming requests that must be validated through the blockchain verification pipeline before being installed in the data plane. For each flow request, the following processing stages occur:

a) Chaincode execution for rule validation
b) Endorsement verification by Fabric peers
c) RAFT-based ordering coordination

Ledger commit and state update

The chaincode validation logic involves rule comparison and hash verification operations and therefore executes in constant time, which is described in Eq. (10):

$$T_{chaincode} = O(1) \tag{10}$$

Endorsement validation, mentioned in the Eq. (11), depends on the number of endorsing peers:

$$T_{endorsement} = O(N_p) \tag{11}$$

Transaction ordering, defined in Eq. (12), through the RAFT consensus protocol requires coordination among ordering nodes:

$$T_{ordering} = O(N_o) \tag{12}$$

Therefore, the computational complexity for verifying a single flow request can be expressed as follows in Eq. (13):

$$T_{flow} = O(N_p+N_o) \tag{13}$$

Overall System Complexity

Across all tenants in the data center, the total number of flow verification requests can be expressed as mentioned in Eq. (14):

$$F_{total} = \sum_{t=1}^{T} F_t \tag{14}$$

The overall computational complexity of the verification framework becomes T_{total}, *which is defined* in Eq. (15):

$$T_{total} = O(F_{total}(N_p+N_o)) \tag{15}$$

Since the count of endorsing peers and ordering service nodes usually is constant in permissioned blockchain systems, the processing cost increases linearly with the number of flow verification requests. This denotes that the suggested framework will be linear when the number of tenants grows.

Computational Overhead: The proposed framework has an overhead that is caused by three major sources:

1. Chaincode execution, which performs flow rule validation and integrity checks
2. Peer endorsement processing, which verifies transaction signatures
3. RAFT-based ordering and ledger commit, which ensure transaction consistency and immutability across Fabric nodes

Although these were on top of these other operations, the results of the experiment show that the proposed architecture can ensure the stability of the transaction latency and moderate resource usage under the growth of multi-tenant flow workloads. It means that the blockchain-based verification mechanism can be introduced into the environment of multi-tenant SDN data centers with manageable overhead and a high level of controller authentication, flow integrity, and tamper-evident auditability guarantees.

6. Limitations of the Proposed Framework

Although the given SDN architecture is based on the concept of blockchain and offers great security guarantees, it has some limitations. First, the implementation of Hyperledger Fabric in control plane of SDN implies the addition of extra overhead, transaction processing overhead based on chain code implementation, endorsement verification and ordering based on RAFT. Even though, the experimental results prove that the latency obtained is within acceptable control-plane operation limits, deploying at very large scales with very high flow rates per transaction can have high commitment delays. Second, the existing execution presupposes the existence of a trusted root controller, which is in charge of controller membership and flow transaction validation. Although such a design makes policy enforcement easier in hierarchical SDN settings, the root controller remains a key component whose failure may affect the control decisions in the entire system. Future research might look at more dispersed models of policy or multi-approver endorsement policy to further disseminate trust.

Third, the testing was carried out in a virtualized multi-node testbed, which is realistic in its simulation of the interaction between the controllers and blockchain but that may not be representative of the scale and distribution of the network in the real deployment of a data center in production. Specialized hardware testbeds or cloud-based deployments would help to offer further knowledge about the scalability of the systems in the real workloads. Last but not least, the proposed structure is geared towards control-plane security and verification of flow integrity as opposed to real-time intrusion detection. Subsequently, the system does not substitute but adds on the existing mechanisms of IDS or anomaly detection. A future research opportunity is to integrate blockchain-supported control-plane validation with more sophisticated techniques of intrusion detection.

7. Conclusion

The synergy of blockchain and hierarchical Software-Defined Networking (SDN) controllers, such as OpenDaylight-Hyperledger Fabric architecture, significantly improves the security, accountability, and transparency. This is so as to minimize such critical security risks as cluster membership without authentication and unscrupulous flow injections using cryptographic on-chain membership validation and two-endorsed flow transactions. The experimental analysis shows that the proposed blockchain-based SDN architecture is feasible in practice. The findings have shown that the system can be able to handle high control-plane workloads without falling down, injection of flows at high rates up to 1000 events per second, and show predictable resource usage with moderate performance. Though it is true that the blockchain validation adds extra transaction commit latency in the face of the traditional SDN controller, the observed overhead is tolerable to control-plane operations. More importantly, such an overhead comes at the cost of the much higher level of security offered by the framework such as unauthorized membership of controller members, malicious flow manipulation, as well as secure coordination of SDN control clusters. Moreover, the impossibility and irreversibility of the blockchain ledger increase the transparency and responsibility of the operations of the controllers. These properties form a reliable control-plane infrastructure that is especially useful in multi-tenant data center models where it is crucial to run the strict control boundaries and assure the isolation between the tenant network domains. Thus, the suggested OpenDaylight-Hyperledger Fabric-based model is highly efficient, scalable and secure model to the latest SDN deployments in the multi-tenant data centers which are able to offer network security posture, operational accountability, and auditing capabilities without harming efficiency or scalability. The next step in work will be the optimization of blockchain transactions processing, their implementation on larger multi-domain scales, and the integration with more sophisticated anomaly detection systems to further improve the security of programmable network structures.

Acknowledgements

This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia (Grant No. KFU263362).

Funding
This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia (Grant No. KFU263362).

Contributions
S.M; S.I; R.S; A.A: Software, Writing-Original Draft, Visualization, Project Administration. S.M: Conceptualization, Methodology, Data Curation, Writing - Review & Editing. S.I: Supervision.

Ethics declarations
This article does not contain any studies with human participants or animals performed by any of the authors.

Consent for publication
Not applicable.

Competing interests
All authors have no conflicts of interest to disclose.

References

[1] Nisar, K., Jimson, E.R., Hijazi, M.H.A., Welch, I., Hassan, R., Aman, A.H.M., Sodhro, A.H., Pirbhulal, S. and Khan, S., 2020. A survey on the architecture, application, and security of software defined networking: Challenges and open issues. Internet of Things, 12, p.100289. https://doi.org/10.1016/j.iot.2020.100289

[2] Lin, C.H., Li, C.Y. and Wang, K., 2018, December. Setting malicious flow entries against SDN operations: attacks and countermeasures. In 2018 IEEE Conference on Dependable and Secure Computing (DSC) (pp. 1-8). IEEE. 10.1109/DESEC.2018.8625101

[3] Hossein, A., Watts, M. and Ahmadi, K., 2019. An overview of multi-controller architecture in software-defined networking. In Proceedings of the CITRENZ Conference (p. 2019).

[4] Bringhenti, D., Yusupov, J., Zarca, A.M., Valenza, F., Sisto, R., Bernabe, J.B. and Skarmeta, A., 2022. Automatic, verifiable and optimized policy-based security enforcement for SDN-aware IoT networks. Computer Networks, 213, p.109123. https://doi.org/10.1016/j.comnet.2022.109123

[5] Bhuiyan, Z.A., Islam, S., Islam, M.M., Ullah, A.A., Naz, F. and Rahman, M.S., 2023. On the (in) security of the control plane of sdn architecture: A survey. IEEE Access, 11, pp.91550-91582. 10.1109/ACCESS.2023.3307467

[6] Alqobaty, A. and Ahmed, N.A.M., 2024. Hybrid architecture for a scalable Data Center Network Based on Blockchain and SDN. مجلة جامعة السعيد للعلوم التطبيقية, 7(1), pp.1-29.

[7] Duy, P.T., Do Hoang, H., Nguyen, A.G.T. and Pham, V.H., 2022. B-DAC: a decentralized access control framework on northbound interface for securing SDN using blockchain. Journal of Information Security and Applications, 64, p.103080. https://doi.org/10.1016/j.jisa.2021.103080

[8] Deng, M., Lyu, Y., Yang, C., Xu, F., Ahmed, M., Yang, N., Xu, Z. and Ke, C., 2024. Lightweight trust management scheme based on blockchain in resource-constrained intelligent IoT systems. IEEE Internet of Things Journal, 11(15), pp.25706-25719. 10.1109/JIOT.2024.3380850

[9] Lakhlef, H., Lerner, T., Kebir, A., El Atia, N., Du, X. and Ingardin, V., 2024, June. Blockchain-Enabled SDN Solutions for IoT: Advancements, Discussions, and Strategic Insights. In 2024 IEEE Symposium on Computers and Communications (ISCC) (pp. 1-6). IEEE. 10.1109/ISCC61673.2024.10733649

[10] Abou El Houda, Z., Hafid, A.S. and Khoukhi, L., 2019. Cochain-SC: An intra-and inter-domain DDoS mitigation scheme based on blockchain using SDN and smart contract. IEEE Access, 7, pp.98893-98907. 10.1109/ACCESS.2019.2930715

[11] Vukolić, M. and Vukoli, M., 2015. The quest for scalable blockchain fabric: proof-of-work vs. BFT replication the quest for scalable blockchain fabric: proof-of-work vs. BFT replication. In Inter-national Workshop on Open Problems in Network. https://doi.org/10.1007/978-3-319-39028-4_9

[12] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J., 2008. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM computer communication review, 38(2), pp.69-74. https://doi.org/10.1145/1355734.1355746

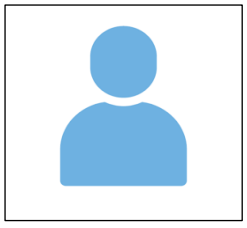
[13] Hu, J., Reed, M., Al-Naday, M. and Thomos, N., 2020, June. Blockchain-aided flow insertion and verification in software defined networks. In 2020 Global Internet of Things Summit (GIoTS) (pp. 1-6). IEEE. 10.1109/GIOTS49054.2020.9119638

[14] Berdik, D., Otoum, S., Schmidt, N., Porter, D. and Jararweh, Y., 2021. A survey on blockchain for information systems management and security. Information Processing & Management, 58(1), p.102397. https://doi.org/10.1016/j.ipm.2020.102397

[15] Krishnamohan, T., 2020. Blockflow: a decentralized sdn controller using block-chain. Theviyanthan Krishnamohan, Kugathasan Janarthanan, Peramune PRLC, Ranaweera AT (2020). 10.29322/IJSRP.10.03.2020.p9991

[16] Sahay, R., Meng, W. and Jensen, C.D., 2019. The application of Software Defined Networking on securing computer networks: A survey. Journal of Network and Computer Applications, 131, pp.89-108. https://doi.org/10.1016/j.jnca.2019.01.019
[17] Yurekten, O. and Demirci, M., 2021. SDN-based cyber defense: A survey. Future Generation Computer Systems, 115, pp.126-149. https://doi.org/10.1016/j.future.2020.09.006
[18] Maruthupandi, J., Prasanna, S., Jayalakshmi, P., Mareeswari, V. and Sanjeevi, P., 2021. Route manipulation aware software-defined networks for effective routing in SDN controlled MANET by disney routing protocol. Microprocessors and Microsystems, 80, p.103401. https://doi.org/10.1016/j.micpro.2020.103401
[19] Xie, R., Cao, J., Li, Q., Sun, K., Gu, G., Xu, M. and Yang, Y., 2022. Disrupting the SDN control channel via shared links: Attacks and countermeasures. IEEE/ACM Transactions on Networking, 30(5), pp.2158-2172. 10.1109/TNET.2022.3169136
[20] Sharma, S., Kumar, A., Bhushan, M., Goyal, N. and Iyer, S.S., 2021. Is blockchain technology secure to work on?. In Blockchain and AI technology in the industrial internet of things (pp. 66-80). IGI Global Scientific Publishing.
[21] Patel, P.B., Thakor, H.P. and Iyer, S., 2019, March. A comparative study on cyber crime mitigation models. In 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 466-470). IEEE.
[22] Schmid, S. and Suomela, J., 2013, August. Exploiting locality in distributed SDN control. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (pp. 121-126). https://doi.org/10.1145/2491185.249119
[23] Blial, O., Ben Mamoun, M. and Benaini, R., 2016. An overview on SDN architectures with multiple controllers. Journal of Computer Networks and Communications, 2016(1), p.9396525. https://doi.org/10.1155/2016/9396525
[24] Wang, H.Z., Zhang, P., Xiong, L., Liu, X. and Hu, C.C., 2016. A secure and high-performance multi-controller architecture for software-defined networking. Frontiers of Information Technology & Electronic Engineering, 17(7), pp.634-646. https://doi.org/10.1631/FITEE.1500321
[25] Badotra, S. and Singh, J., 2017. Open Daylight as a Controller for Software Defined Networking. International Journal of Advanced Research in Computer Science, 8(5).
[26] Suh, D., Jang, S., Han, S., Pack, S., Kim, T. and Kwak, J., 2016, June. On performance of OpenDaylight clustering. In 2016 IEEE NetSoft Conference and Workshops (NetSoft) (pp. 407-410). IEEE. 10.1109/NETSOFT.2016.7502476
[27] Lara, A., Kolasani, A. and Ramamurthy, B., 2013. Network innovation using openflow: A survey. IEEE communications surveys & tutorials, 16(1), pp.493-512. 10.1109/SURV.2013.081313.00105
[28] Enns, R., 2006. NETCONF configuration protocol (No. rfc4741).
[29] Moriarty, K.M., 2020. Transforming Information Security: Optimizing Five Concurrent Data Trends to Reduce Resource Drain. Emerald Publishing Limited. https://doi.org/10.1108/9781839099281
[30] Ferraiolo, D., Cugini, J. and Kuhn, D.R., 1995, December. Role-based access control (RBAC): Features and motivations. In Proceedings of 11th annual computer security application conference (pp. 241-48).
[31] Mohanta, B.K., Panda, S.S. and Jena, D., 2018, July. An overview of smart contract and use cases in blockchain technology. In 2018 9th international conference on computing, communication and networking technologies (ICCCNT) (pp. 1-4). IEEE. 10.1109/ICCCNT.2018.8494045
[32] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y. and Muralidharan, S., 2018, April. Hyperledger fabric: a distributed operating system for permissioned blockchains. In Proceedings of the thirteenth EuroSys conference (pp. 1-15). https://doi.org/10.1145/3190508.319053
[33] Ravi, N. and Shalinie, S.M., 2021. BlackNurse-SC: A novel attack on SDN controller. IEEE Communications Letters, 25(7), pp.2146-2150. DOI: 10.1109/lcomm.2021.3075898
[34] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M. and Kumar, D., 2017. Understanding the mirai botnet. In 26th USENIX security symposium (USENIX Security 17) (pp. 1093-1110).
[35] Hu, J., Reed, M., Thomos, N., AI-Naday, M.F. and Yang, K., 2020. Securing SDN-controlled IoT networks through edge blockchain. IEEE Internet of Things Journal, 8(4), pp.2102-2115. https://doi.org/10.1109/JIOT.2020.3017354
[36] Mozumder, A.H. and Basha, M.J., 2025. SmartSecChain-SDN: A Blockchain-Integrated Intelligent Framework for Secure and Efficient Software-Defined Networks. arXiv preprint arXiv:2511.05156. 10.14445/23488549/IJECE-V12I10P117
[37] Garg, S., Goyal, S. and Bhandari, A., 2025. A lightweight blockchain based scalable and collaborative mitigation framework against new flow DDoS attacks in SDN enabled autonomous systems. Scientific Reports, 15(1), p.36002. https://doi.org/10.1038/s41598-025-19989-2

Biographies



Stefina Macwan. is a cybersecurity researcher specializing in Software-Defined Networking (SDN), blockchain-based trust frameworks, and security analytics. Her research focuses on securing OpenDaylight controller architectures using Hyperledger Fabric for tamper-proof logging, access control, and network auditability. She has hands-on experience with SDN testbeds, intrusion detection systems, SIEM-based threat hunting, and cyber-range platforms. Her work integrates zero-trust networking, micro-segmentation, and

blockchain-backed network forensics to strengthen critical network infrastructures. Her long-term objective is to design verifiable and resilient SDN security architectures for large-scale enterprise and national cyber-defense environments. stefinamacwan@gmail.com

Dr. Sailesh Suryanarayan Iyer holds a Ph.D. in Computer Science and a Post-Doctoral Fellowship from the University of Louisiana at Lafayette, USA, and currently serves as Professor and In-Charge Principal at NSIT-IFSCS, Ahmedabad. He has over 24 years of experience across academia, industry, and corporate training, including more than 20 years in core academic leadership. He has successfully supervised nine Ph.D. scholars and serves as Co-Guide for Post-Doctoral researchers at international universities in the USA, Singapore, and Malaysia. He is a recipient of the Research Excellence Award for three consecutive years (2021–2023) and serves as an Honorary Adjunct Research Scientist at Neurolabs International, Dana Brain Health Institute, Iran. He holds multiple granted and published patents and is actively involved as an editor with Elsevier, Wiley, CRC Press (Taylor & Francis), IGI Global, and Bentham Science. His research interests include computer vision and image processing, cybersecurity, data mining and analytics, artificial intelligence, machine learning, and blockchain technologies. drsaileshiyer@gmail.com